CLAIMS

What is claimed is:

1. A method for building a program for execution by a programmable controller, the method comprising:

receiving a source program, the source program including high-level instructions for controlling a programmable controller;

converting the source program to a first processor-executable program and a second processor-executable program;

comparing the first and second processor-executable programs;

sending, in response to the first and second processor-executable programs being substantially the same, one of the first and second processor-executable programs to the programmable controller.

2. The method of claim 1 wherein the sending includes sending the one of the first and second processor-executable programs to one of a plurality of intelligent field devices, the intelligent field devices including programmable controllers.

3. The method of claim 1 wherein the sending includes sending the one of the first and second processor-executable programs to a modular redundant control system, the modular redundant control system including at least two processor modules, wherein the modular redundant control system distributes the one of the first and second processor-executable programs to the at least two processor modules, and wherein each of the processor modules is configured to carry out instructions encoded in the one of the first and second processor-executable programs.

4. The method of claim 1 wherein the first processor-executable program and the second processor-executable program include encoded instructions for controlling field devices.

5. The method of claim 1 wherein the first processor-executable program and the second processor-executable program include operating-system instructions for controlling the programmable controller.

6. The method of claim 1 including:

loading a first compiler into a first memory location; and

loading a second compiler into a second memory location separate from the first memory location;

wherein the converting includes converting the source program to a first piece of intermediate code with the first compiler and converting the source program to a second piece of intermediate code with the second compiler.

7. The method of claim 6 wherein the first and second memory locations are a part of the same computer.

8. The method of claim 6 wherein the first and second compilers are simultaneously loaded into the first and second memory locations.

9. The method of claim 6 wherein the converting includes storing the first and second pieces of intermediate code in separate disk drives.

10. The method of claim 1 wherein the comparing includes comparing the first and second processor-executable programs bit-by-bit.

11. The method of claim 1 wherein the comparing includes generating a cyclical redundancy code for each of the first and second processor-executable programs, and comparing the cyclical redundancy codes for the first and second processor-executable programs.

12. The method of claim 1 wherein the converting includes converting the source program to the first processor-executable program with a first processor and converting the source program to the second processor-executable program with a second processor.

13. The method of claim 1 wherein the converting includes converting the source program to the first processor-executable program and asynchronously converting the source program to the second processor executable program.

14. The method of claim 1 including:

receiving a reference source program and a reference executable program, wherein the reference executable program includes processor executable code generated on a trusted system from the reference source program;

converting the reference source program to a test program using system resources used to convert the source program to the one of the first and second processor-executable programs;

comparing the reference executable program with the test program; and

validating, in response to the reference executable program and the test program being substantially the same, the system resources used to convert the source program to the one of the first and second processor-executable programs.

15. A system for generating a control program comprising:

a processor;

a network communication module coupled to the processor, wherein the network communication module is configured to communicate with a programmable controller via a network, the programmable controller associated with a safety instrumented function; and

a memory associated with the processor, the memory including:

conversion code including encoded instructions for separately converting a source program into first and second load modules, the first and second load modules including processor-executable code to control the programmable controller; and

comparison code including encoded instructions for comparing the first and second load modules and sending one of the load modules to the programmable controller in response to the first and second load modules being substantially the same.

16. The system of claim 15 wherein the programmable controller is an intelligent field device, and wherein at least one other intelligent field devices is coupled to the network.

17. The system of claim 15 wherein the programmable controller is a main processor in a modular redundant control system, wherein the sending includes sending the one of the load modules to at least one other programmable controller in the modular redundant control system.

18. The system of claim 15 wherein the one of the load modules includes encoded instructions for controlling field devices with the programmable controller.

19. The system of claim 15 wherein the one of the load modules includes operating-system instructions for controlling the programmable controller.

20. The system of claim 15 wherein the conversion code includes encoded instructions for simultaneously generating the first and second load modules using the processor.

21. The system of claim 20 including:

random access memory coupled with the processor;

wherein the conversion code includes encoded instructions to load two copies of a portion of the conversion code into separate portions of the random access memory and simultaneously generate each of the first and second load modules with a corresponding one of the two copies of the portion of the conversion code.

22. The system of claim 15 wherein the conversion code includes encoded instructions to asynchronously generate the first and second load modules using the processor.

23. The system of claim 15 including another processor, wherein the conversion code includes encoded instructions to generate the first load module with the processor and the second load module with the other processor.

24. The system of claim 15 including:

a first storage device coupled to the processor;

a second storage device coupled to the processor;

wherein the conversion code includes encoded instructions to store intermediate code associated with the first load module in the first storage device and store intermediate code associated with the second load module in the second storage device.

25. The system of claim 24 wherein the first and second storage devices are hard drives.

26. The system of claim 24 wherein the first storage device is a local hard drive and the second storage device is a network hard drive.

27. The system of claim 15 wherein the comparison code includes encoded instructions to compare the first and second load modules on a bit-by-bit basis.

28. The system of claim 27 wherein the comparison code includes encoded instructions for sending the one of the load modules to the programmable controller in response to the first and second load modules being identically the same.

29. The system of claim 15 wherein the comparison module includes instructions to generate cyclical redundancy codes for the first and second load modules, and instructions to compare the cyclical redundancy codes for the first and second load modules.

30. The system of claim 15 wherein the memory includes:

a reference source program; and

a reference executable program;

wherein the conversion code includes encoded instructions to convert the reference source program into a test program using system resources associated with the one of the load modules sent to the programmable controller;

wherein the comparison code includes encoded instructions for comparing the test program and the reference executable program, and validating, in response to the test program and the reference executable program being substantially the same, the system resources associated with the one of the load modules sent to the programmable controller.

31. A processor-readable medium including code to generate an executable program from a source program when carried out by a processor, the executable program being disposed for execution by a programmable controller, the code comprising:

conversion code for converting a source program into the executable program;

a code segment for loading two copies of at least a portion of the conversion code into a memory associated with the processor such that the two copies do not occupy the same location in the memory; each of the two copies including code to generate a corresponding one of two copies of at least a portion of the executable program;

comparison code for comparing the two copies of the at least the portion of the executable program, and for sending at least one of the two copies of the at least the portion of the executable program to the programmable controller in response to the two copies of the at least the portion of the executable program being substantially the same.

32. The processor-readable medium of claim 31 wherein the conversion code includes compiler code for converting the source program into intermediate code, and wherein the code for loading two copies of the at least the portion of the conversion code into the memory includes code for loading two copies of the compiler code into the memory such that the two copies of the compiler code do not occupy the same location in memory.

33. The processor-readable medium of claim 32 including:

code for loading a first piece of intermediate code generated by one of the two copies of the compiler code into a first memory location and loading a second piece of intermediate code generated by another one of the two copies of the compiler code into a second memory location.

34. The processor-readable medium of claim 33 wherein each of the first and second memory locations are a part of a corresponding one of a first and second hard drives.

35. The processor-readable medium of claim 34 wherein each of the first and second hard drives are controlled by separate disk controllers.

36. The processor-readable medium of claim 33 wherein each of the first and second memory locations are a part of a corresponding one of a first and second separate RAM memory locations.

37. The processor-readable medium of claim 33 wherein the first memory location is part of a network hard drive and the second memory location is part of a memory location selected from the group consisting of a RAM memory and a local hard drive.

38. The processor-readable medium of claim 31 wherein each of the two copies of at least a portion of the conversion code are in a corresponding one of two dynamic loaded libraries (DLLs), wherein the code for loading the two copies includes code for loading the two copies into the memory simultaneously such that the two copies do not occupy the same location in the memory.

39. The processor-readable medium of claim 31 wherein the comparison code includes code for comparing the two copies of the at least the portion of the executable program on a bit-by-bit basis.

40. The processor-readable medium of claim 31 wherein the comparison code includes:

a code segment for generating a cyclical redundancy code for each of the two copies of the at least the portion of the executable program; and

a code segment for comparing the cyclical redundancy codes for the two copies of the at least the portion of the executable program.

41. The processor-readable medium of claim 31 including:

a reference source program;

a reference executable program, the reference executable program includes processor executable code generated on a trusted system from the reference source program;

wherein the conversion code includes code for converting the reference source program to a test program using system resources used to convert the at least one of the two copies of the at least the portion of the executable program sent to the programmable controller;

wherein the comparison code includes code for comparing the reference executable program with the test program and validating, in response to the reference executable program and the test program being substantially the same, system resources used to convert the at least one of the two copies of the at least the portion of the executable program sent to the programmable controller.

42. A method for building a program for execution by a programmable controller, the method comprising:

receiving a source program, the source program including high-level instructions for controlling a programmable controller, the programmable controller being associated with a safety instrumented function;

converting the source program to a first processor-executable program and a second processor-executable program;

comparing the first and second processor-executable programs;

generating a signal indicative of the similarity between the first and second processor-executable programs.

43. The method of claim 42 including:

loading a first compiler into a first memory location; and

loading a second compiler into a second memory location separate from the first memory location;

wherein the converting includes converting the source program to a first piece of intermediate code with the first compiler and converting the source program to a second piece of intermediate code with the second compiler.

44. The method of claim 43 wherein the first and second memory locations are a part of the same computer.

45. The method of claim 43 wherein the first and second compilers are simultaneously loaded into the first and second memory locations.

46. The method of claim 43 wherein the converting includes storing the first and second pieces of intermediate code in separate disk drives.

47. The method of claim 42 wherein the first processor-executable program and the second processor-executable program include encoded instructions for controlling field devices associated with the safety instrumented function.

48. The method of claim 42 wherein the first processor-executable program and the second processor-executable program include operating-system instructions for controlling the programmable controller.

5